Short and smooth trajectories, formally proved

Yves Bertot
`Yves.Bertot@inria.fr`

The STAMP team is designing a program to compute smooth trajectories for robots between obstacle described by line segments (which can be joined to form polygons). The current version of the algorithm produces trajectories that are unnecessarily wavy and and sometimes take long detours.

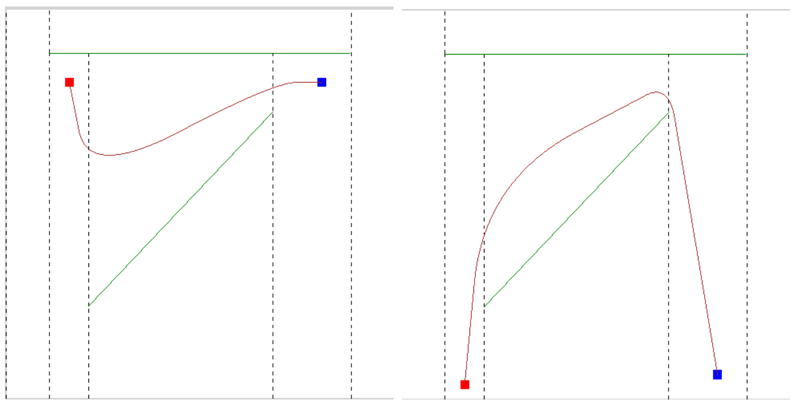

Figure 1: Two examples of un satisfactory trajectories

The objective of this internship is to modify the algorithm so that :

- straight line trajectories are used in priority when this is relevant
- optimally short paths are preferred, at least as an intermediate step before smoothing the trajectories

The objective is not so much to produce extremely competitive algorithms as to produce algorithms that are proved correct. We expect the proofs to be performed with Coq [1] and the Mathematical Components library [4], and the final algorithm to be extracted and integrated in a web-page for demonstration purposes.

During this formal proof task, the considered algorithm will receive as input a collection of cells that are known to be collision free. The formal proof will need to verify that produced trajectories are included in the cells, or more directly that the trajectories themselves are collision free. For the optimality criterion, it is foreseen that such a property will be out of reach, but ways to express approximate optimality will be studied.

Context: Proofs will be performed using the Coq system[1] and the Mathematical Components library[2]. The supervisor and his team will provide access to computers with Coq and the relevant libraries installed and training. Instructions to produce demonstrations and animations with the formally verified code will be provided by the STAMP team.

[1]`https://coq.inria.fr`
[2]`http://math-comp.github.io/math-comp/`

Prerequisites: The pre-requisite for this internship is a good knowledge of functional programming.

Tasks: The intern will have to write various implementations of the algorithm, either using plain inductive data structures, or using data-types for finite sets and graphs provided in the Mathematical Components library. They will also have to write specifications expressing the required safety properties for the output. They will then have to perform proofs, mostly using the Coq system and the existing theorems of the Mathematical Components library [4].

References

[1] Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development, Coq'Art: The Calculus of Inductive Constructions. Springer. 2004.

[2] Latombe, J.-C.: Robot Motion Planning. Kluwer Academic Publishers. 1991.

[3] LaValle, S. M.: Planning Algorithms. Cambridge University Press. 2006. `http://planning.cs.uiuc.edu/`

[4] Mahboubi A., Tassi E.: Mathematical Components. To appear. `https://math-comp.github.io/mcb/`